DDoS Vulnerability Analysis of Bittorrent Protocol

Ka Cheung Sia
kcsia@cs.ucla.edu

Abstract

Bittorrent (BT) traffic had been reported to contribute to 30% of the Internet traffic nowadays and the number of participants have been growing rapidly. For such a protocol that significantly involved in the Internet traffic, the robustness and security must be evaluated carefully. In this paper, we analyze the BT protocol and identify several potential vulnerabilities available for malicious Internet users to exploit and leverage them and launch Distributed Denial of Service (DDoS) attack. We demonstrate such possibility by launching a DDoS attack using one of the methods studied and reveal some measurements recorded on such attack. We then propose fixes to the existing BT protocol and discuss critical issues when designing a peer-to-peer (P2P) network in the future.

1 Introduction

Bittorrent (BT) [2] traffic had been reported to contribute to 30% of the Internet traffic nowadays and the number of participants have been growing rapidly. From the port history reported by SANS Internet Storm Center [9], port 6881, which is a commonly used BT port number, is always in the top 10 list of port being scanned in the Internet. For such a widely used protocol in the Internet, its robustness and security must be evaluated carefully.

As BT users, we have experienced traffic similar to DDoS attack occasionally when a popular BT seed is used. In certain scenarios, it had been recorded that more than 1000 clients are trying to connect to a particular host simultaneously. Such a burst of connection attempts chokes the host to be incapable of both sending and receiving legitimate traffic (which include the BT traffic itself). Although the host behaves as a legitimate participant within the peer-to-peer(P2P) BT network and downloads only one file share, the amount of traffic generated is enough to stop the host from running properly. We believe this may infer a vulnerability in the BT protocol for Distributed Denial (DDoS) attackers and may require special attentions to avoid possible attacks by this mean.

Under the current BT protocol, clients discover and communicate with each others by first communicating with trackers, which act as servers to host and exchange information about the file segment locations. By providing tempered information to the trackers or hosting compromised trackers, it is possible to redirect huge amount of BT traffic to a victim under attack. In this work, we make the following contributions to provide a more secured use of P2P network:

- We study the BT protocol and identify vulnerabilities for launching a DDoS attack.
- We ran DDoS experiments, that statistics recorded in the victim side shows that such vulnerability can be devastating.
- We propose fixes to the current BT protocol and implementations and enumerate security issues against DDoS attack when designing future P2P network.

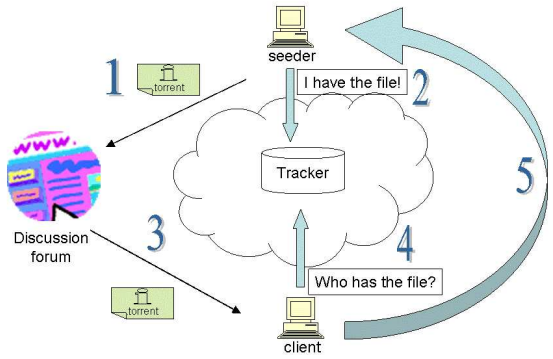

Figure 1: Illustration of sharing files in BT network.

In Section 2, we give an overview of the BT protocol and analyze some existing vulnerabilities. We detail the procedure of a DDoS attack in Section 3 and give measurements to project the traffic under the scenario when a victim is under serious attack. In Section 4, we propose the fixes to the current BT protocol and implementations and discuss issues to be addressed when designing a more secure-aware P2P network in the future. We give overview of current related research finally in Section 5

2 BT protocol study and exploit

BT is a P2P file sharing protocol. Unlike other popular P2P file sharing network protocols such as Gnutella, KaZaa, etc., it separates the file discovery function from the P2P network, making the network serve only the purpose of exchanging file content. Figure 1 illustrate five key steps for an individual to share files in the BT network.

1. A peer first generates a corresponding torrent file for the files he wants to share, which consists of names of the files to be shared, hashes of the file content, trackers to be used, etc. (we will explain the detail of these afterwards). It then publishes this torrent file through some other online communication channels such as newsgroups, discussion forums, or some Websites dedicated for publishing and announcing torrent files.
2. The initiating peer then notifies the tracker that it is sharing the files described in the torrent he has just generated.
3. Another peer who is also interested in the shared files looks up the torrent file from the online communication channels and downloads to its local computer.
4. This torrent file contains necessary information for a BT client program to initiate a connection to the tracker in use. It then asks the tracker for a list of other peers who are currently sharing the files.
5. It initiates connections to other participating peers and start requesting pieces of the files until it finises. At the same time, if a file piece is completed, it will start serving other peer's request for that piece.

The torrent file contain hashes of pieces of the file, in which a piece is usually of size 128KB, 256KB, 512KB, ... By doing this, a peer can request multiple pieces of a file from different

peers so that a peer can start sharing the content even when it is only partially completed. Also, the hashes are used to verify the validity of every piece upon finishing the download. A peer uses two methods in BT protocol to know which peers to contact and request for file pieces, namely tracker and trackerless.

- *tracker* - There is a dedicated machine that keeps track of which computers are downloading and uploading file pieces specified in the torrent file, every peer will ask for information from this tracker. The URL of the tracker being used is stored in the torrent file and it can run over HTTP or UDP protocol.
- *trackerless (or DHT)* - A group of computers participating in the DHT network (this network is different from the BT file sharing network) share the responsibility of the tracker, i.e. every node in the DHT network act as the tracker for a certain torrent files. DHT lookup method is used to resolve the location of the node who is serving as the tracker of a file share.

For the tracker protocol running over HTTP, a client sends a GET request to the tracker URL, indicated in the torrent file, to announce that it is sharing the files, the tracker then returns a list of other peers who are currently sharing the files in the BT network. The GET message takes the following input parameters.

- `info_hash` - A 20 byte SHA1 hash of the content in the torrent file to identify the file share itself.
- `peer_id` - A random string of 20 bytes generated by a peer to identify itself.
- `ip` - An optional parameter giving the IP of the participating peer.
- `port` - The port number a peer is listening on, the default one is 6881.
- `uploaded` - The total number of bytes a peer has uploaded so far.
- `downloaded` - The total number of bytes a peer has downloaded so far.
- `left` - The number of bytes a peer has to download before completing the task.
- `event` - It can be `started`, `completed`, or `stopped` to indicate the status of the peer. If this value is missing, it serves as a keep-alive message saying that the peer is still in the progress of downloading.

Clearly, the IP address and port number can be altered to point to the victim's IP address and the port it is listening on. The tracker will then broadcast this information to the peers in the BT network and the rest of the BT clients will attempt to connect to the victim. The tracker protocol over UDP works similarly and has the same vulnerability for a peer to specify the IP address in the announce message. Although whether to accept the `ip` input parameter varies across different tracker implementation, in the later experiment, a successful DDoS attack shows that there exists a significant number of trackers in the Internet that accept this input parameter.

The trackerless protocol runs over UDP, and it no longer allows a peer to specify the IP address in the `ANNOUNCEMENT` message. Instead, the source address of the UDP `ANNOUNCEMENT` packet is used by default. It is not possible to spoof the source address as the protocol is lightly authenticated, where the `ANNOUNCEMENT` needs to specify the token value previously received from the tracker node in the `GET_PEER` message. Although it is not possible to spoof the `ANNOUNCEMENT` message, there exists another vulnerability in the `PING` message used in the DHT network. A node sends out `PING` messages to notify other nodes that it is participating in the DHT network. By spoofing the source address of this `PING` message, one is able to redirect huge DHT query traffic towards a victim machine. We identify the vulnerability here and for the readers who are interested in exploiting this, the detail protocol specification can be found at [1].

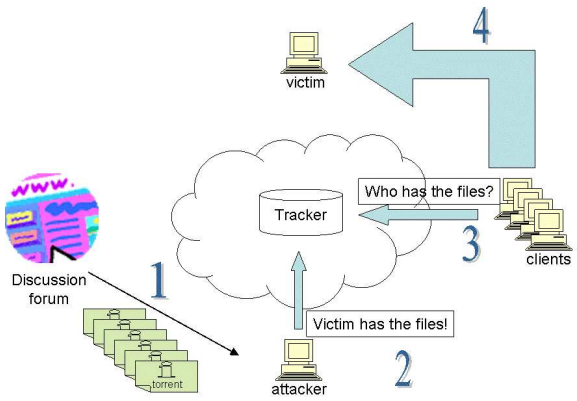

Figure 2: Illustration of the key steps in an attack.

Number of torrents	1191
Corrupted	6
Torrents using 1 tracker	999
Torrents using multiple trackers	186
Torrents supporting DHT	121
Number of trackers	2048 (1963 http, 85 udp)
Number of unique http trackers	311
Number of unique udp trackers	21

Table 1: Summary of torrents used

3 Experiment

Among the vulnerabilities described in the previous section, we choose to send tempered information to trackers. The steps are shown in Figure 2.

1. We download 1191 recently uploaded torrent files from `http://www.mininova.org`, which is a Website dedicated to share torrent files among users. A summary of the torrents and trackers used are listed in Table 1.
2. The original python BT client program is modified to parse the torrent files and send forged announcement message to the corresponding trackers indicated in each torrent file.
3. Upon the trackers receive requests for a list of participating peers from other clients, it will send them the victim's IP address and port number.
4. Other peers in the BT network will then attempt to connect to the victim machine and request for pieces of files.

Our first experiment is to show the feasibility of redirecting traffic to a victim machine. For every torrent file, we contact the trackers once announcing that the victim (131.179.187.205) has started sharing on port 80. The victim machine is running an Apache web server configured to serve 400 clients simultaneously. It is also running two monitoring programs: `netstat` and `tcpdump` to track the number of new and concurrent TCP connections. On the left of Figure 3, it shows that half an hour after the attack begins, the victim machine established around 500

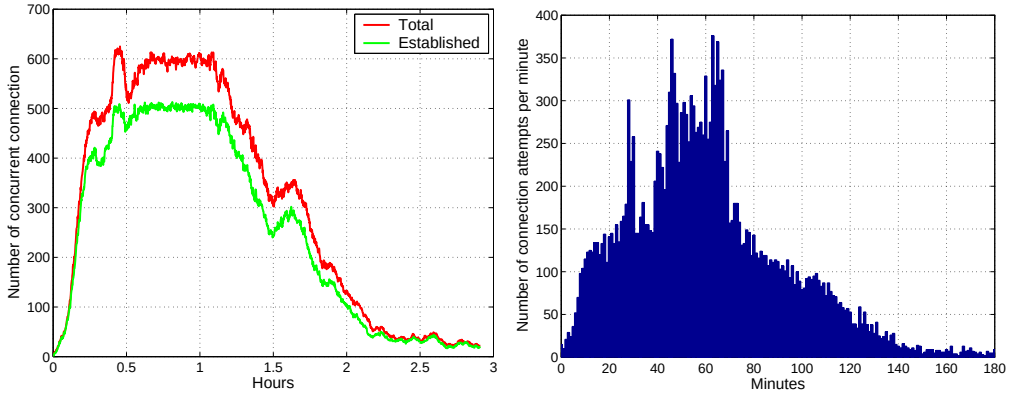

Figure 3: Number of concurrent connections and connection attempts during the attack.

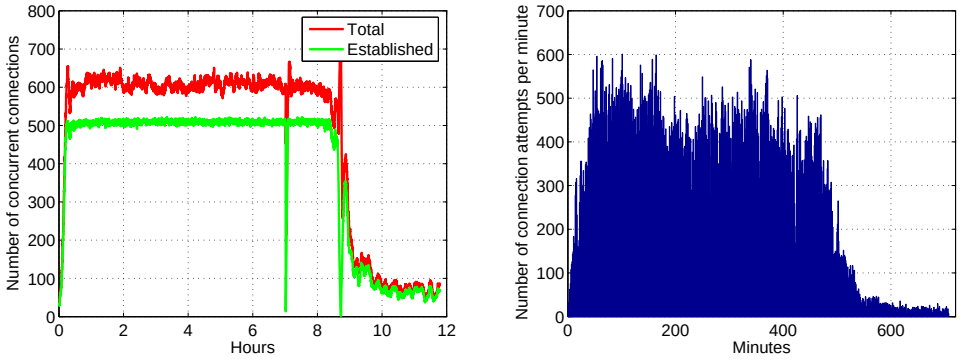

Figure 4: Number of concurrent connections and connection attempts during the larger scale attack.

concurrent TCP connections, while on the right, it shows that at the peak of this attack, the victim receives about 250 connection requests per minutes.

The intention of the previous experiment is to show the feasibility of redirecting traffic to victim machine. The second experiment will show whether such attack can sustain a long period of time and consume enough resource from the victim to become a DDoS attack. This time, for every torrent, we sent 10 announcements[1] to the trackers over a period of 8 hours. On the left of Figure 4, it shows that the victim maintains 500 concurrent TCP connections throughout the attack period, while on the right, it shows that it receives on average 450 connection attempts per minute. At the same time, the web server either timeout the connection or give heavy delay when other legitimate clients (web browsers) attempt to retrieve the front page. During the attack period, connections to other service of the victim (such as SSH) were able to go through, this indicates that the attack can be classified as an application level resource attack rather than a bandwidth attack on the victim.

There were 30,513 distinct IPs attempted to connect to the victim. The distribution of the number of connection attempts made by each IPs is shown in the left of Figure 5. It shows that most BT clients attempted connection a few times over this period; while a significant number of them tried three times, this probably indicates that common BT clients will retry connection

[1]The first one is a `started` message and the rest are keep-alive messages.

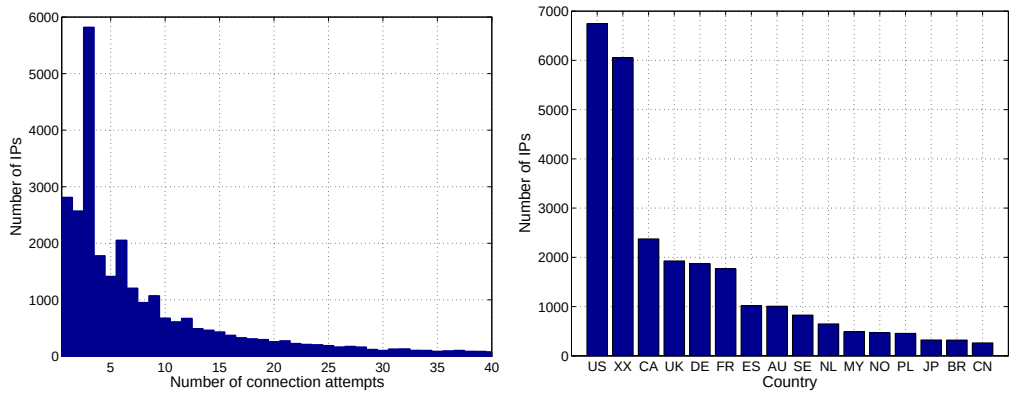

Figure 5: Distribution of number of connection attempts and IP's geographical location.

three times before giving up when they are not running the same protocol as the other side. Two IPs (202.156.6.67 from Singapore and 24.22.183.141 from the United States) were recorded to connect more than 8000 times during this period. Such high number of connection attempts suggests that they can possibly be agents hired by the music and movie industry doing content pollution [4] or other researchers performing experiments on the BT network at the same time. In addition, the right of Figure 5 shows the the geographical location distribution of all the connecting IPs, it indicates that such attack is distributed globally and is hard to guard against by ingress filtering at upstream routers.

4 Discussion of solution

We have investigated the DDoS vulnerability of the current BT protocol; given the significant use of BT protocol in the Internet traffic, its robustness and security must be evaluated carefully. Clearly, allowing an user to arbitrarily specify its own IP make such protocol vulnerable to DDoS attackers. This can be fixed by a straighter tracker implementation which only uses the source IP of a connection, or improve the tracker protocol that authenticate the user with its source address such as the scheme used in DHT trackerless protocol. However, both measures require Internet-wide deployment to be effective; as long as there is a portion of the BT network contains the vulnerabilities, they can still be exploited for DDoS attack.

Another option is to impose packet filtering in upstream router of the victim. Since a network operator knows which type of service the downstream machine is providing, takes HTTP web server for example, packets going to port 80 of the machine should contain certain characteristics of HTTP request. Figure 4 illustrates the scenario when a BT client initiates a connection to our victim web server; the DDoS attack use up the resource of the web server because every BT client establishes a full TCP connection to the victim. The well-known SYN-cookie technique can guard against typical SYN flood attack but not in the scenario we illustrated because the third packets in a connection setup is never being received at the victim side for SYN-flood attack. Using similar idea, we may filter out the third packet sent by a BT client and reuse the SYN-cookie technique. Normally, the third packet in a TCP connection setup by a legitimate HTTP client will contain the HTTP request header in the payload; while if the other side is a BT client, the payload will contain BT protocol request header. If an upstream router is able to filter out the third packet based on its payload content together with the signature in the TCP header, the victim will never establish a full TCP connection and can again be safe guarded by the SYN-cookie technique.

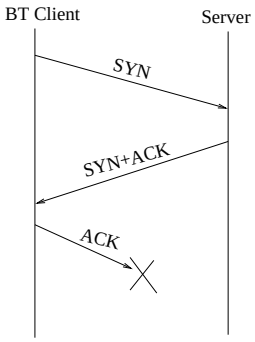

Figure 6: A possible solution to BT DDoS attack.

5 Related work

In [7], Mirkovic and Reiher give a detail survey on different DDoS attack and defense mechanisms. The exploit using BT traffic described in this paper roughly falls into the genre of reflector attack and SYN flood to use up application resource. The exploit described in this paper further verify the theoretical work by Douceur [3] that any systems allow an individual to act as many roles as she wants, they are suspectable to a DDoS attack to certain extent.

There are other work that also try to exploit P2P network. Liogkas et.al. [6] illustrate that by behaving as a non-cooperative BT client, one can take advantages of other BT clients to fasten its download time and reduce bandwidth usage, etc.. Liang et.al. [5] described an index poisoning attack in P2P file sharing systems, which is essentially announcing false piece of files to stop the network from functioning properly. Recently, Naoumov and Ross [8] study the Overnet protocol and identify two vulnerabilities to launch a DDoS attack. The study is similar to this paper; however, up to our best knowledge, this work is the first to address issues on exploiting BT traffic to harm external systems.

6 Conclusion

In this project, we have studied the BT protocol and identify its vulnerabilities for DDoS attackers. We have exploited one of them and demonstrated the severeness of launching DDoS attack using these popular P2P network. Based on the observation during the attack, we identify both a proactive solution (changing of BT protocol and tracker implementation) and a passive solution (filtering at the victim side) to safe guard against such kind of attack.

References

[1] Bram Cohen. Bittorrent dht protocol extension draft. `http://www.bittorrent.org/Draft_DHT_protocol.html`.

[2] Bram Cohen. BitTorrent protocol specification. `http://www.bittorrent.org/protocol.html`.

[3] John R. Douceur. The Sybil Attack. In *International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[4] J. Liang, R. Kumar, Yongjian Xi, and Ross. Pollution in P2P File Sharing Systems. In *Proceedings of INFOCOM 2005*, 2005.

[5] Jian Liang, Naoum Naoumov, and Keith W. Ross. The Index Poisoning Attack in P2P File Sharing Systems. In *IEEE Conference on Computer Communication*, Barcelona, Spain, April 2006.

[6] Nikitas Liogkas, Robert Nelson, Eddie Kohler, and Lixia Zhang. Exploiting BitTorrent For Fun (But Not Profit). In *International Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.

[7] Jelena Mirkovic and Peter Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *Computer Communications Review*, 34(2), April 2004.

[8] Naoum Naoumov and Keith Ross. Exploiting P2P Systems for DDoS Attacks. In *Proceeding of the International Workshop on Peer-to-Peer Information Management*, Hong Kong, May 2006.

[9] SANS. SANS - Internet Storm Center.